HELDA
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

UNIVERSITY OF HELSINKI

https://helda.helsinki.fi

An ARM cluster for running CMSSW jobs

Osmani, Lirim

2020-11-16

Osmani , L & Linden , T 2020 , ' An ARM cluster for running CMSSW jobs ' , EPJ Web of Conferences , vol. 245 , 05038 . https://doi.org/10.1051/epjconf/202024505038

http://hdl.handle.net/10138/326245
https://doi.org/10.1051/epjconf/202024505038

cc_by
publishedVersion

Downloaded from Helda, University of Helsinki institutional repository.

This is an electronic reprint of the original article.

This reprint may differ from the original in pagination and typographic detail.

Please cite the original version.

EPJ Web of Conferences **245**, 05038 (2020) https://doi.org/10.1051/epjconf/202024505038
CHEP 2019

An ARM cluster for running CMSSW jobs

Lirim Osmani[1,*] and *Tomas* Lindén[2,**]

[1]Department of Computer Science, University of Helsinki, PB 68 (Gustaf Hällströmin katu 2b), FI-00014 University of Helsinki, Finland
[2]Helsinki Institute of Physics, PB 64 (Gustaf Hällströmin katu 2), FI-00014 University of Helsinki, Finland

Abstract. The ARM platform extends from the mobile phone area to development board computers and servers. It could be that in the future the importance of the ARM platform will increase for High Performance Computing/High Throughput Computing (HPC/HTC) if new more powerful (server) boards are released. For this reason Compact Muon Solenoid Software (CMSSW) has previously been ported to ARM in earlier work.
The CMSSW is deployed using the CERN Virtual Machine File System (CVMFS) and the jobs are run inside Singularity containers. Some ARM AArch64 CMSSW releases are available in CVMFS for testing and development. In this work CVMFS and Singularity have been compiled and installed on an ARM cluster and the AArch64 CMSSW releases in CVMFS have been used. We report on our experiences with this ARM cluster for CMSSW jobs.
Commodity hardware designed around the 64-bit architecture has been the basis of current virtualization trends with the advantage to emulate diverse environments for a wide range of computational scenarios. However, in parallel the mobile revolution have given a rise to ARM SoCs with primary focus on power efficiency. While still in the experimental phase, the power efficiency and 64-bit heterogeneous computing already point to an alternative option for traditional x86_64 CPUs servers for datacenters.

1 Introduction

To meet the computing challenge posed by the High-Luminosity Large Hadron Collider (HL-LHC) improved software performance, changed analysis procedures and new hardware resources are required [1]. ARM computers are developed for the highly comptetive mobile phone market, so they might provide less expensive computing resources for HL-LHC computing.

2 Hardware

CMSSW has been ported in earlier work to armv7 [2, 3] and to armv8 [4, 5]. The available ARM systems are usually either servers or inexpensive development boards. The ARM architecture is also coming to HPC systems. The successor of the Japanese K-supercomputer,

*e-mail: lirim.osmani@helsinki.fi
**e-mail: tomas.linden@helsinki.fi

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

Table 1. Comparison of the Odroid-N2 and the Raspberry Pi 4 Model B.

Board	Odroid-N2	Raspberry Pi 4 Model B
CPU	Amlogic S922X	BCM2711B0
Line width	12 nm	28 nm
Clock frequency	1.8 GHz	1.5 GHz
	1.9 GHz	
Cores	4 x Cortex A73	4 x Cortex A72
	2 x Cortex A53	
L1 cache	32 kB	32 kB
L2 cache	1 MB + 256 kB	1 MB
GPU	Mali-G52	VideoCore VI
RAM	2, 4 G	2, 4, 8 GB
RAM speed	DDR4 1320	LPDDR4-3200 SDRAM
Ethernet	1 Gb/s	1 Gb/s
USB	4 x USB 3.0	2 x USB 3.0, 2 x USB 2.0
	1 x USB 2.0 OTG	1 x USB C OTG
μSD	UHS-I	DDR50
eMMC	5.1	
Board size	90 x 90 mm^2	85 x 56 mm^2

named Fugaku will be built using 7 nm Fujitsu A64FX chips. The first-generation European Processor Initiative chip family, named Rhea, will include ARM ZEUS architecture general purpose cores for HPC applications. In the server market Amazon uses the 7 nm Graviton2 chip which is based on ARM Neoverse N1 cores. Also the 7 nm Ampere Altra chip based on ARM Neoverse N1 cores is coming to the server market. These new chips are potentially interesting from the perspective of HL-LHC computing.

In this work we used small ARM development boards, two hexa core Odroid-N2 [6] from HardKernel and two quad core Raspberry Pi 4 Model B (RPi4 in the following) [7] from the Raspberry Pi Foundation because these two have 4 GB RAM available with recent ARM cores. Unfortunately we did not have access to any ARM servers. Both boards have been bought with 4 GB of RAM, which is more than the 1–2 GB found on earlier similar boards, but less than the 2 GB / core required by the Compact Muon Solenoid (CMS) LHC experiment for standard grid sites. The machines can still be used for many tasks. A comparison of the details of the ARM boards is shown in Tab. 1.

The Odroid-N2 bottom is covered by a heat sink providing enough cooling to avoid thermal throttling of the CPU during prolonged high load usage.

The RPi4 CPU is equipped with a metallic heat spreader, which keeps the CPU temperature below the throttling limit of 80 °C only for brief high load bursts. In order to have maximum performance and consistent benchmark results the cooling was improved on one RPi4 4 B with a 30x30x7 mm fan and on the other one with a 32x32x20 mm heat sink. Both the fan and the heat sink increased the cooling power enough that thermal throttling can be avoided when running the *stress-ng* program on four cores for three hours. All systems were run with their default clock frequencies to have reproducible results because the overclocking potential is strongly dependent on the variable quality of each silicon die and on the performance of the cooling system.

On all boards 128 GB of flash storage was used, eMMC on the Odroid-N2s and μSD on the Pi4s with a six GB large swapfile.

3 Operating system

The Odroid-N2 was released with AArch64 userspace available on Ubuntu 18.04 LTS with kernel 4.9.187-53, so it was used. At the time of this work Ubuntu and Gentoo supported the AArch64 userspace and kernel on the RPi4, so Ubuntu 18.04.3 LTS with a Gentoo kernel was used until Ubuntu 19.10 was released. Initially on Ubuntu 19.10 we used kernel 5.3.0-1008-raspi2. Before the multicore runs the kernels were upgraded to 4.9.210-66 on the Odroid-N2 and 5.3.0-1015-raspi2 on the RPi4.

4 Software environment

The used software environment consists of the following software components. Singularity is a HPC container technology for scientific applications with reproducible, sharable and distributed features [8]. Singularity was used to run a CERN CentOS 7 (CC7) AArch64 Docker image, because CMSSW is supported on the 64-bit CC7 platform.

Singularity, developed since 2015 at the Lawrence Berkeley National Lab (LBNL) is a container system developed for scientific research and high-performance computing applications. Singularity is the default container technology in the CMS experiment. Contrary to Docker, Singularity does not aim to create completely isolated environments. It relies on a more open model with the objective of providing integration with existing tools installed on the host operating system. Consequently, the only namespace that is isolated between the host and a singularity container is the file system. Other namespaces remain untouched by default. Thus, the network stack, process tree, and user space are the same between container and host, which leads to the container being seen as a process which is executed in the host operating system. This feature is very important for two reasons. First, Singularity containers can be started and killed by any tool used to manage processes, such as *mpirun* or Slurm. Second, because the user space is untouched, the user that executes processes inside the container is the same as the one which has started the container, which means that regular users can start a container without needing root access in the host OS.

Moreover, Singularity also allows the users to leverage the resources of whatever host you are on. This includes HPC interconnects, resource managers, file systems, GPUs and/or accelerators, etc. Singularity does this by enabling several key facets:

- encapsulation of the environment;
- containers are image based;
- no user contextual changes or root escalation allowed;
- no root owned daemon processes.

CVMFS [9] is a scalable software distribution service to assist High Energy Physics (HEP) collaborations to deploy their software across different environment and sites. CVMFS is also used outside of the HEP community. The AArch64 compiled CVMFS was used as a software distribution service for CMSSW. The development AArch64 CMSSW releases available in CVMFS were used for the actual application.

5 Cluster software

Our cluster testbed runs on resources at the Computer Science Department of the University of Helsinki. It is a hybrid cluster setup consisting of AArch64 and Intel x86_64 machines for running CMSSW jobs. The cluster software stack consist of Advanced Resource Connector (ARC) 6 as the standard grid interface [10] and HTCondor as the batch system [11].

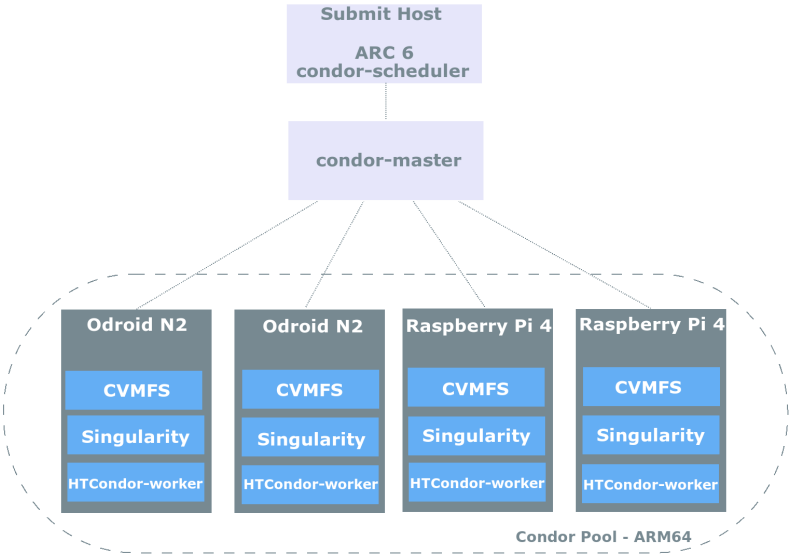

Figure 1. The ARM software environment.

The ARC 6 submit host has CentOS 7 on an Intel Core i5 with 8GB of RAM with 2 network interfaces. It also contains the HTCondor scheduler. The rest of HTCondor components are configured on the HTCondor-master node that runs Ubuntu 18.04 LTS on another Intel Core i5 with 8GB of RAM configured with a single private network interface . Our AArch64 worker pool consists of two Odroid-N2 and two Raspberry Pi 4 units, see Fig 1.

6 Power measurements

The RPi4 is powered by a 5 V / 3 A power supply with a USB-C connector. The power drawn by the RPi4 board itself was measured with an AVHzY CT-2 USB-power meter for the single core jobs, neglecting the losses of the power supply.

The currents for the multicore jobs on the RPi4, the Odroid-N2 and a Kaby Lake x86_64 reference system were measured with an Agilent (presently Keysight) E3634A power supply over a serial connection with the Keysight BenchVue program.

The power usage of the RPi4 is being optimized by the manufacturer with improved firmware versions. At the time of this work Ubuntu 19.10.1 had a VideoCore VI firmware dated Aug 15 2019. The bootloader version dated Sep 10 2019 and the USB controller VL805 firmware version 000137ab were used.

The runtimes and energy consumption depends on the kernel, operating system, container image, firmware versions and on the CVMFS cache state. The very first *runTheMatrix* run was used only to populate the CVMFS cache without recording the data. The idle power and using a synthetic load with *stress-ng* is shown in Tab. 2.

7 Performance

The performance of the storage I/O and memory systems on the studied computers can quickly be studied with the hdparm tool. It is not an accurate benchmark, but it gives an

Table 2. Power measurements.

Computer	Idle	*stress-ng*
Odroid-N2	2.1±0.1 W	5.4±0.1 W
Raspberry Pi 4 Model B with heat sink	2.6 ±0.1 W	5.8±0.1 W
Kaby Lake 14 nm quad core i7-7700 32 GB RAM Fedora 29	8±1 W	99±1 W

Table 3. Average of three hdparm runs.

Computer	Cached reads	Buffered disk reads
Kaby Lake SSD	18408 MB/s	381 MB/s
Odroid-N2 eMMC	2124 MB/s	144 MB/s
Raspberry Pi 4 Model B μSD	1098 MB/s	43 MB/s

Table 4. ROOT compilation time and ROOTMARKS.

Computer	Compilation time	ROOTMARKS
Kaby Lake	1 h 2 min 7 s	4999.9
Odroid-N2	2 h 7 min 33 s	1158.8
Raspberry Pi 4 Model B	3 h 51 min 39 s	911.0

Table 5. Runtime and energy consumption of the CMSSW runs.

Computer	Average runtime	Energy / J
Kaby Lake	4 min 28 s	7265
Odroid-N2	17 min 13 s	2896
Raspberry Pi 4 Model B	22 min 0 s	4262

estimate of the read performance of a storage system and also the memory and cache system bandwidth, see Tab. 3.

ROOT release 6.18/04 [12] was compiled with six cores on an Odroid-N2, with four cores on a RPi4 and with four cores on the x86_64 system. The compilation time is shown in Tab. 4. A swap partition had to be added to be able to compile on the ARM boards with several cores. The *stressHepix* benchmark was run three times on each system and the average ROOTMARKS results are shown in Tab. 4.

The CMSSW software validation and data quality monitoring tool *runTheMatrix* was installed and the additional needed modules were compiled. Local *runTheMatrix* jobs were run with these commands :

```
singularity shell -B /cvmfs docker://cmssw/cc7:aarch64
export SCRAM_ARCH=slc7_aarch64_gcc700
source /cvmfs/cms.cern.ch/cmsset_default.sh
cmsrel CMSSW_10_2_0_pre6
cd CMSSW_10_2_0_pre6
cmsenv
time runTheMatrix.py -l 2.0 -job-reports -j 4
```

The average runtime of three single threaded runs is shown in Tab. 4 and the corresponding estimated energy consumption. *Both of the ARM boards used significantly less energy for this computation than the x86_64 system, see Fig 2.*

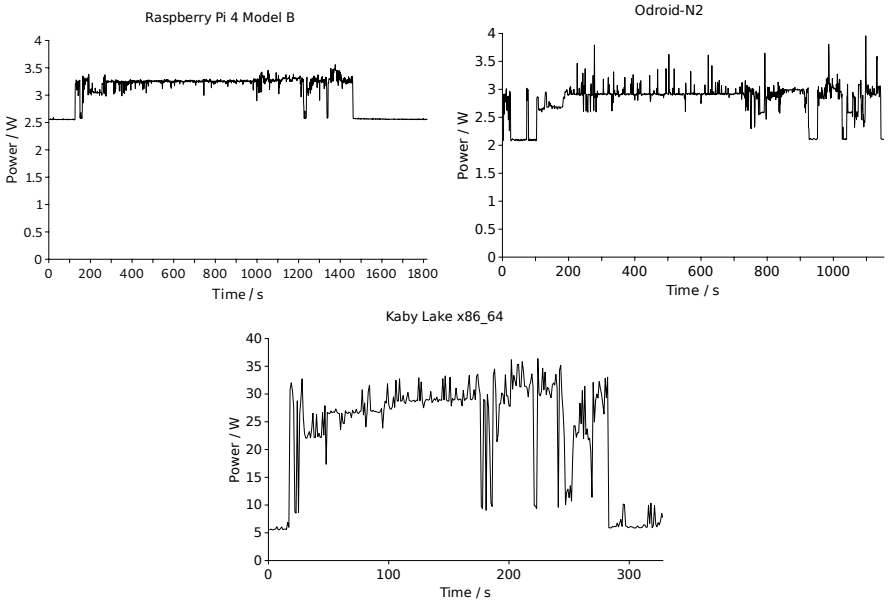

Figure 2. The RPi4 power during a CMSSW job averaged 3.23 W and that of the Odroid-N2 2.82 W and on a Kaby Lake 27.14 W.

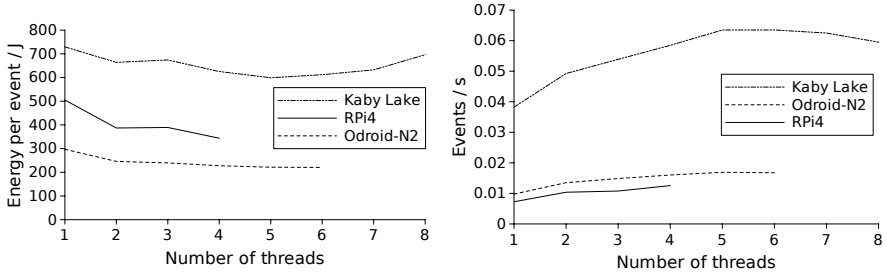

Figure 3. The energy / event and processed events / s as a function of number of threads for the ARM boards.

Multithreaded CMSSW [13] runs were also done while recording the execution time and measuring the current with a sampling frequency of 1 Hz. CMSSW multithreading parallelizes events and modules within the events. It can support modules also doing parallel operations internally, but this is not actually used in production at this point. The multithreaded jobs were run once for 1..4 threads on the RPi4, for 1..6 threads on the Odroid-N2 and for 1..8 threads on the quad core Kaby Lake system with hyperthreading enabled, see Fig. 3. The used workflow runs event generation, which is not multithreaded, digitization and reconstruction. Scaling by the number of threads can therefore only be expected to be seen for the two later tasks. The x86_64 system has 32 GB of RAM, which is significantly more than the 4 GB on the ARM systems. About 7 MB of swap was taken into use on the Odroid-N2 when running with 6 threads, so the 4 GB of RAM is not a big limiting factor for the used workflows on the ARM systems. The ARM boards use significantly less energy per

processed event compared to the Kaby Lake system. The Kaby Lake system is faster than the ARM boards in terms of events processes per second and it scales better also.

8 Summary

In this work we have explored the viability of the ARM architecture for CMS workloads by compiling and installing the essential software stack. ROOT, Singularity and CVMFS were compiled successfully on two kinds of AArch64 ARM boards, the Odroid-N2 and the Raspberry Pi 4 Model B. The CMSSW software stack can be run in Singularity on these ARM boards using an AArch64 CERN CentOS 7 Docker image. The ARM boards have been used as the worker nodes of a hybrid cluster testbed using ARC 6 and HTCondor.

These computers draw 2–3 W when idling and about 6 W with synthetic CPU load. Both of the ARM boards used to run CMSSW using *runTheMatrix* used significantly less energy than the x86_64 system.

For the tested applications the Odroid-N2 is faster than the RPi4. The x86_64 architecture executes faster the used workloads than both of the ARM systems, but with a significantly higher energy cost. The main advantage of the ARM systems is the significantly lower energy usage per calculation compared to the x86_64 system as shown by our results.

9 Funding and acknowledgements

This work was jointly funded by Doctoral School in Computer Science (DoCS) and NODES research lab at the University of Helsinki. This work was also supported by the Otto A. Malm foundation.

We gratefully acknowledge the help of Juha Aaltonen, Sami Lehti and Tor Paulin.

References

[1] J. Albrecht et al., Comput Softw Big Sci (2019) 3, 7
[2] D. Abdurachmanov et al, J. Phys.: Conf. Ser. **523** 012009 (2014)
[3] D. Abdurachmanov et al, J. Phys.: Conf. Ser. **513** 052008 (2014)
[4] D. Abdurachmanov et al, J. Phys.: Conf. Ser. **608** 012032 (2015)
[5] D. Abdurachmanov et al, J. Phys.: Conf. Ser. **608** 012033 (2015)
[6] https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram/
[7] https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
[8] G.M. Kurtzer, V. Sochat V, M.W. Bauer, PLoS ONE **12(5)**: e0177459 (2017) https://doi.org/10.1371/journal.pone.0177459. See also https://sylabs.io/singularity/
[9] J. Blomer et al, J. Phys.: Conf. Ser. **396** 052013 (2012), see also https://cernvm.cern.ch/portal/filesystem
[10] M. Ellert et al., Future Generation Computer Systems **23** 219-240 (2007), see also http://www.nordugrid.org/arc/arc6/
[11] D. Thain, T. Tannenbaum, and M. Livny, Concurrency and Computation: Practice and Experience, Vol. **17**, No. 2-4, pages 323-356 (2005), see also https://research.cs.wisc.edu/htcondor/.
[12] R. Brun and F, Rademakers, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A **389** 81-86 (1997), see also http://root.cern.ch/
[13] C.D. Jones and E. Sexton-Kennedy J. Phys.: Conf. Ser. 513 022034 (2014)